The National Virtual Observatory Book
ASP Conference Series, Vol. 382, © 2008
M. J. Graham, M. J. Fitzpatrick, and T. A. McGlynn, eds.

Chapter 19: Web-based Tools—Montage: An Astronomical Image Mosaic Engine

G. Bruce Berriman, J. C. Good and A. C. Laity

J. C. Jacob and D. S. Katz

E. Deelman, G. Singh, and M.- H. Su

Introduction

Montage (`http://montage.ipac.caltech.edu`) is a toolkit for aggregating astronomical images into mosaics. Its scientific value derives from three features of its design:

- It preserves the calibration and astrometric fidelity of the input images to deliver mosaics that meet user-specified parameters of projection, coordinates, and spatial scale. It supports all projections and coordinate systems in use in astronomy.

- It contains independent modules for analyzing the geometry of images on the sky, and for creating and managing mosaics; these modules are powerful tools in their own right and have applicability outside mosaic production, in areas such as data validation. It can take advantage of SIAP services to discover input images that meet the specifications of the sky coverage of the output mosaic.

- It is written in ANSI-compliant C, and is portable and scaleable – the same engine runs on desktop, cluster or supercomputer environments running common Unix-based operating systems such as Linux, Solaris, Mac OS X and AIX.

It was developed specifically to provide a grid-enabled image mosaic engine for the NVO. The code is available for download for non-commercial use from `http://montage.ipac.caltech.edu/docs/download.html`. The current distribution (version 3.0) includes the image mosaic processing modules and executives for running them, utilities for managing and manipulating images, and all third-party libraries, including the FITS library and WCS tools. The distribution also includes modules for installing Montage on computational grids. A web-based Help Desk is available to support users, and documentation is available on-line, including the specification of the API and installation and set –up requirements.

Montage is highly scaleable. It uses *the same set of modules* to support two instances of parallelization: MPI (Message Passing Interface; `http://www-unix.mcs.anl.gov/mpi/`), a library specification for message passing, and Pegasus (Planning and Execution for Grids), a toolkit that maps workflows onto distributed

processing environments (Deelman et al. 2005). Parallelization and performance are described in detail at `http://montage.ipac.caltech.edu/docs/grid.html` and in Katz et al. (2005).

Montage is in active use in generating science data-products, in underpinning quality assurance and validation of data, in analyzing scientific data and in creating Education and Public Outreach products (`http://montage.ipac.caltech.edu/applications.html`). A number of research groups are exploiting the scalability of Montage as a driver application to optimize the performance of scheduling algorithms (Blythe et al. 2005; Zhang et al. 2006), portal development (Singh et al. 2005), and workflow managers (Truong and Fahringer 2005).

This chapter describes the design of Montage and walks though a tutorial that shows how to produce a mosaic.

1. Montage Architecture and Algorithms

This section summarizes the Montage architecture and algorithms. Berriman et al. (2002, 2004) and the online user guide at `http://montage.ipac.caltech.edu/docs/` provide more detail.

1.1. Supported File Formats

Montage supports two-dimensional images that adhere to the definition of the FITS standard. (Calabretta and Greisen 2002; and `http://fits.gsfc.nasa.gov/fits_home.html`). In FITS, the relationship between the pixel coordinates in the image and physical units is defined by the *World Coordinate System* (WCS). Included in the WCS is a definition of how celestial coordinates and projections are represented in the FITS format as *keyword=value* pairs in the file headers. Montage analyzes these pairs of values to discover the footprints of the images on the sky and calculates the footprint of the image mosaic that encloses the input footprints. Montage supports all projections supported by WCS, and all common astronomical coordinate systems. The output mosaic is FITS-compliant, with the specification of the image parameters written as keywords in the FITS header. The toolkit contains a utility that will convert the FITS image to a JPEG format image.

1.2. Design Philosophy

There are four steps in the production of an image mosaic:

- Discover the geometry of the input images on the sky from the input FITS keywords and use it to calculate the geometry of the output mosaic on the sky

- Re-project the input images to the spatial scale, coordinate system, WCS-projection, and image rotation.

- Model the background radiation in the input images to achieve common flux scales and background level across the mosaic.

- Co-add the re-projected, background-corrected images into a mosaic.

Each production step has been coded as an independent engine run from an executive script. This toolkit design offers flexibility to users. They may, for example, use Montage as a re-projection tool, or deploy a custom background rectification algorithm while taking advantage of the re-projection and coaddition engines.

1.3. Algorithms Used in Montage

Image re-projection Image re-projection redistributes the flux from a set of input pixels to a set of output pixels. Montage preserves the photometric and positional (astrometric) integrity of the input images by projecting both input and output pixels onto the celestial sphere, as shown in Fig 1. This approach reduces re-projections to a problem in classical spherical trigonometry, that of calculating the area of overlap of two convex polygons on a sphere (with no further consideration of the projections themselves).

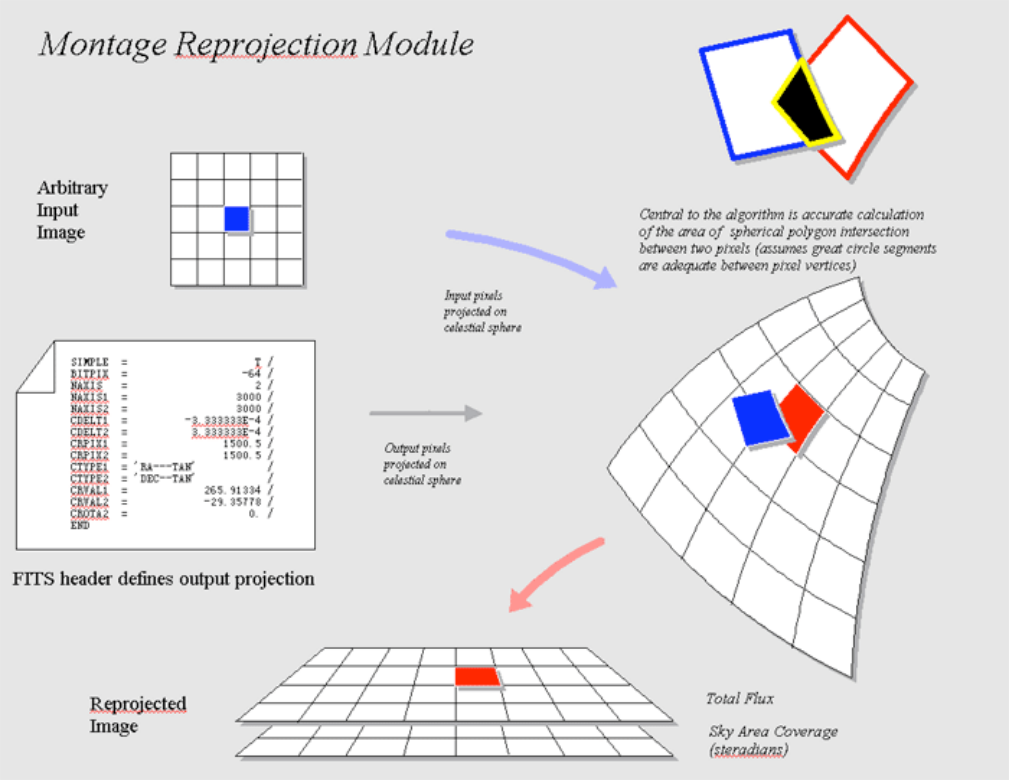

Fig 1: The General Re-projection Algorithm Used By Montage

General algorithms exist for determining the overlap of polygons in Cartesian space (O'Rourke 1998). A modification of this approach for use in spherical coordinates determines the intersection polygon on the sphere (a convex hull) and applies

Girard's Theorem, which gives the area of a spherical triangle based on the interior angles, to calculate the polygon's area. For any two overlapping pixels, the algorithm yields the area of the sky from the input pixel that contributes energy to the output pixel. This provides a mechanism for accurately distributing input energy to output pixels and a natural weighting mechanism when combining overlapping images. This re-projection algorithm also supports the Drizzle algorithm from STScI, a method for the linear reconstruction of an image from under-sampled, dithered data (Fruchter and Hook 2002).

The generality in the re-projection algorithm just described comes at the expense of speed (e.g. serial re-projection of 54 2MASS images, covering 1 square degree of sky, on a 2.3 GHz Linux processor with 1 GB memory takes 5500 seconds). There is, however, a technique that can improve performance by a factor of 30 under specific circumstances. When input images are represented as tangent projections, the geometry of the system can be viewed as a pair of gnomonic (tangent plane, TAN) projection planes intersecting the coordinate sphere (see Figure 2). A single line connects the center of the sphere, the projected point on the first plane and the projected point on the second plane. This geometric relationship results in transformation equations between the two planar coordinate systems that require no trigonometry or extended polynomial terms. This approach excludes many common projections such as Cartesian and zenithal equidistant, and is essentially limited to small areas of few square degrees. Processing of all-sky images, as is almost always the case with projections such as Aitoff, generally requires the slower plane-to-sky-to-plane approach.

An extension of this technique applies to images of high resolution and relatively small extent (up to a few degrees on the sky). It involves approximating the projection by a "distorted" TAN projection. In this case, the pixel locations are distorted by small distances relative to the plane used in the image projection formulae. A *distorted space* is one in which the pixel locations are slightly offset from the locations on the plane used by the projection formulae, as happens when detectors are slightly misshapen. This distortion is modeled by pixel-space polynomial correction terms that are stored as parameters in the image FITS header.

While this approach was developed to deal with physical distortions caused by telescope and instrumental effects, it is applicable to Montage in augmenting the plane-to-plane re-projections. Over a small, well-behaved region, most projections can be approximated by a TAN projection with small distortions. For instance, in terms of how pixel coordinates map to sky coordinates, a two-degree Cartesian projection is identical to a TAN projection with a fourth-order distortion term to within a percent of a pixel width.

Background Modeling and Rectification The Montage background matching algorithm assumes that terrestrial and instrumental backgrounds can be described by simple functions or surfaces (e.g. slopes and offsets). That is, the local background has essentially all its energy at the lowest spatial frequencies. This approach is clearly inapplicable when the background varies at the same frequencies as the astrophysical signal; background modeling will then require custom approaches that depend on the properties of the data set.

Given a set of overlapping images, characterization of the overlap differences is key to determining how each image should be adjusted before combining them. Montage takes the approach of considering each image individually with respect to its neighbors. Specifically, we determine the areas of overlap between each image and

its neighbors, and use the complete set of overlap pixels in a least-squares fit to determine how each image should be adjusted (e.g. what gradient and offset should be added) to bring it "best" in line with its neighbors.

The net effect is to subtract a low-frequency (currently a gradient/offset) background from each image in such a way that the cumulative image-to-image differences are minimized. To speed the computation (and minimize memory usage), we approximate the gradient and offset values by a planar surface fit to the overlap area difference images rather than perform a least-squares fit.

Coadditions and Weighting of Output Pixel Each input pixel's contribution to the flux in a mosaic pixel is added to that pixel weighted by the sky area of the overlap. In addition, a cumulative sum of these sky area contributions is kept for the output pixels (essentially and physically an "area" image). When combining multiple overlapping images, the output pixel flux is simply the average of the images being combined, weighted by the sky area of the overlap.

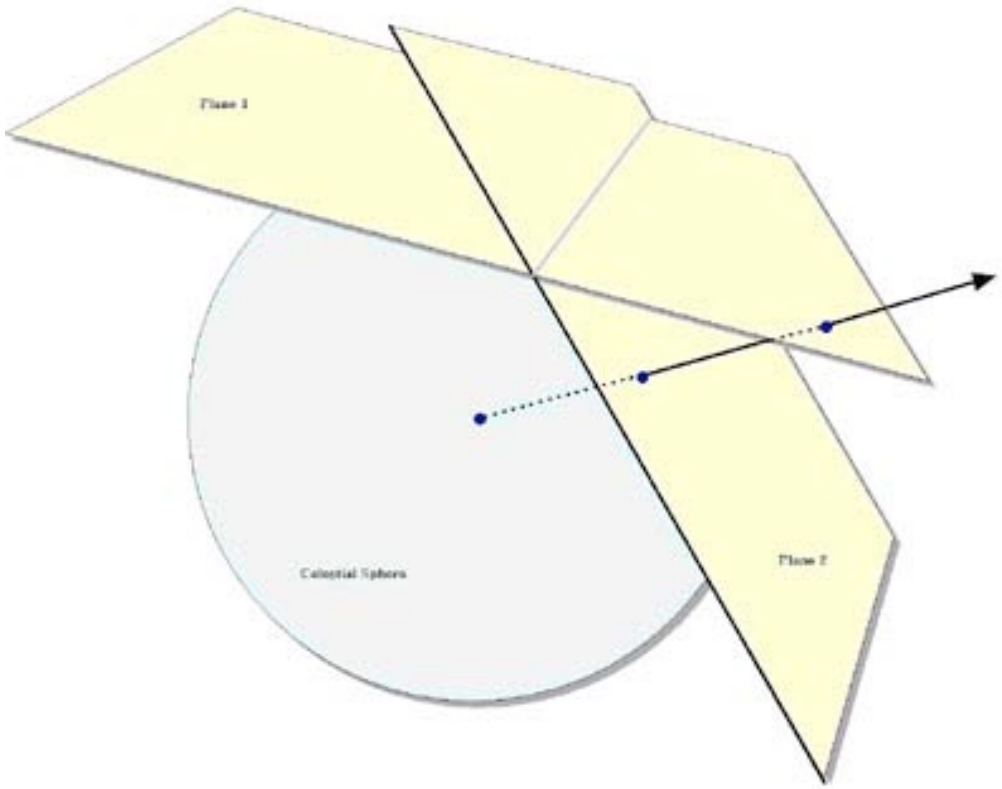

Fig 2: Plane-to-plane Re-projections For Gnomonic transformations

The limitations of available memory have been simply overcome in coaddition determining by reading the re-projected images one line at a time from those files that contribute flux to each pixel. The coaddition module accumulates "stacks" of

input pixel values and area coverage value. The contents of the output row are then calculated one output pixel (i.e. one input stack) at a time, by averaging the flux values from the stack.

Montage currently supports mean and median coaddition, with or without weighting by area. The mean algorithm (default) accumulates flux values contributing to each output pixel, and then scales them by the total area coverage for that pixel. The median algorithm ignores any pixels whose area coverage falls below a specific threshold, and then calculates the median flux value from the remainder of the stack.

Accuracy The calibration and positional fidelity of the algorithms have been investigated by applying the *Sextractor* source extraction tool (Bertin and Arnouts 1996) to mosaics created from synthetic images containing point sources superposed on variable sky backgrounds. When biases involving measurement of fluxes and positions are taken into account, the fluxes of 99.7% of the sources are within 0.1% of the original value, and all positions lie within 0.1 pixels of the original positions. These results apply to a wide range of projections, coordinates, sampling rates and image rotations. These results have been independently verified by the Spitzer Wide-Area Infrared Experiment (SWIRE) project. It performed similar experiments with simulations of images that simulate the infrared sky as seen by the Infrared Array Camera (IRAC) aboard the Spitzer Space Telescope. Their report is available online at `http://montage.ipac.caltech.edu/docs/swirevalidation.html`.

2. How To Use The Montage Toolkit

2.1. Montage Use Case: Creating a Three-Color Mosaic of 2MASS and MSX Images.

This section shows how to use Montage to create a 3-color mosaic of the War and Peace nebula (NGC 6537). The mosaic aggregates 170 J and K_s images that cover 3 square degrees of the 2MASS survey, and an image at 8 μm that covers 2.4 square degrees of the Galactic Plane survey of the Midcourse Space Experiment (MSX). (`http://irsa.ipac.caltech.edu/Missions/msx.html`). Table 1 summarizes in order of use those Montage modules used in composing the mosaic in Figure 3 from the input images. This use case is an abbreviated version of the tutorial described in Laity et al. (2005). Users are urged to visit the Montage web page for updates to this walkthrough.

In the walkthrough, each Montage command is followed by the output of the module, which is always a structured output message to `stdout`, of the form `[struct stat="{\it status}", key1="{\it val1}", ... , keyn="{\it valn}"]`.

Step One: Setup metadata tables and output header templates

Read the image metadata from the FITS keywords and write them to a column delimited ASCII table for the Montage modules to read:

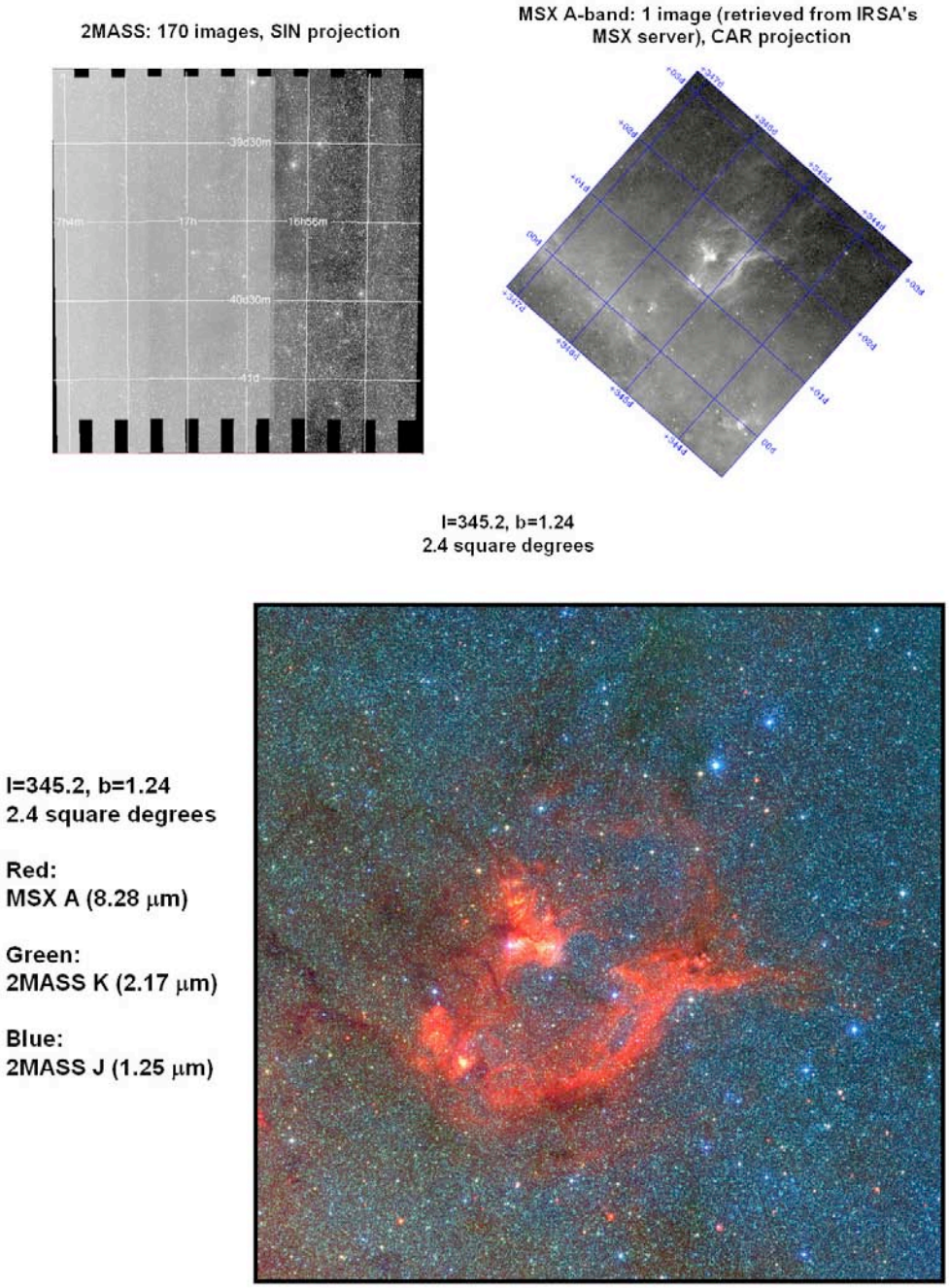

Fig 3: Creation of a three-color mosaic from 2MASS and MSX data. The top two images show the input images, and the bottom image shows them aggregated into a three-color mosaic.

```
> mImgtbl raw_K raw_K.tbl
[struct stat="OK", count=170, failed=0, nooverlap=0]
```

Call `mMakeHdr` to calculate and footprint on the sky that completely encloses all the 2MASS images, and write this to a template file:

```
> mMakeHdr raw_K.tbl template.hdr
[struct stat="OK", count=170, clon=254.587292, clat=-40.25175
3, lonsize=2.353611, latsize=2.450000, posang=359.891421, lon
1=256.154189, lat1=-41.468162, lon2=253.014309, lat2=-41.4636
21, lon3=253.076184, lat3=-39.014964, lon4=256.104469, lat4=
-39.019343
```

Step Two: Image re-projection.

The 2MASS images are in a TAN projection, and can use the fast re-projections module. To speed up the MSX re-projections, create a distorted TAN header for the MSX data:

```
> mGetHdr raw_MSX/msx_4deg.fits msx.hdr
[struct stat="OK", ncard=23]
> mTANHdr -c eq msx.hdr msxtan.hdr
[struct stat="OK", fwdxerr=0.00351429, fwdyerr=0.0054
6297, fwditer=51, revxerr=0.00335636, revyerr=0.03825
81, reviter=9
```

Launch the 2MASS image re-projections by calling the re-projections executable, `mProjExec`, and using the "`-f`" flag to instruct it to use fast re-projections:

```
> mProjExec -f -p raw_K raw_K.tbl template.hdr proj_K stats_K.tbl
[struct stat="OK", count=170, failed=0, nooverlap=0]
```

Next, call the fast re-projections module directly on the MSX data, instructing it to use the distorted TAN template header instead of reading the native MSX FITS header:

```
> mProjectPP -i msxtan.hdr raw_MSX/msx_4deg.fits  final_MSX.fits
template.hdr
[struct stat="OK", time=6082]
```

Once the re-projections are complete, regenerate the image metadata tables, as the FITS geometry has changed:

```
> mImgtbl proj_K proj_K.tbl
[struct stat="OK", count=170, badfits=0]
```

Step Three: Background rectification.

The background rectification process matches each image's background to its surrounding images, globally minimizing the inter-image differences. First create a table that lists the pairs of files that overlap and identifies each pair:

```
> mOverlaps proj_K.tbl diff_K.tbl
[struct stat="OK", count=454]
```

Then create "difference" images for each overlap region by subtracting FITS files from each other:

Table 1: Montage Modules Used In Creating The 2MASS-MSX Mosaic

Component	Description
`mImgtbl`	Extract geometry information from a set of FITS headers and create a metadata table from it.
`mMakeHdr`	Returns the FITS header for the mosaic to be generated from a list of input images.
`mGetHdr`	Reads FITS image header and prints to text file
`mTANHdr`	Analyzes a template file and determines if there would be an adequate equivalent distorted TAN projection
`mProjExec`	A simple executive that runs `mProject` for each image in an image metadata table.
`mProjectPP`	Performs a plane-to-plane re-projection on a FITS input image
`mOverlaps`	Analyze an image metadata table to determine which images overlap on the sky.
`mDiffExec`	Run `mDiff` on all the overlap pairs identified by `mOverlaps`.
`mFitExec`	Run `mFitplane` on all the `mOverlaps` pairs. Creates a table of image-to-image difference parameters.
`mBgModel`	Modeling/fitting program which uses the image-to-image difference parameter table to interactively determine a set of corrections to apply to each image to achieve a "best" global fit.
`mBgExec`	Run `mBackground` on all the images in the metadata table
`mAdd`	Coadd the reprojected images to produce an output mosaic.
`mJPEG`	Generates a JPEG image file from a FITS file (or a set of three FITS files in color).

```
> mDiffExec -p proj_K diff_K.tbl template.hdr diff_K
[struct stat="OK", count=454, failed=0]
```

Call `mFitplane` on each difference image to find the plane that best fits each one:

```
> mFitExec diff_K.tbl fits_K.tbl diff_K
[struct stat="OK", count=454, failed=0, warning=0, missing=0]
```

Using the information found by `mFitplane`, calculate which planes need to be removed from each image to globally minimize the background differences:

```
> mBgModel proj_K.tbl fits_K.tbl corrections_K.tbl
[struct stat="OK"]
```

Finally, call `mBackground` on each projected image to subtract the plane calculated by `mBgModel`:

```
> mBgExec -p proj_K proj_K.tbl corrections_K.tbl corr_K
[struct stat="OK", count=170, nocorrection=0, failed=0]
```

Step Four: Coaddition

The last step for the 2MASS data is to co-add them into mosaics:

```
> mAdd -e -p corr_K proj_K.tbl template.hdr final_K.fits
[struct stat="OK", time=144]
```

Step Five: Creating a 3-color JPEG image

After cropping the edges out of each mosaic, call `mJPEG` to create a 3-color JPEG image from the 3 FITS files. The user assigns a color to each image and inputs the desired color-stretch, which can be found using a visualization tool such as the IRSA OASIS tool at `http://irsa.ipac.caltech.edu/applications/Oasis/`. The output of `mJPEG` is the bottom image in Figure 3.

```
> mJPEG -red final_MSX_crop_4.fits 0% 99.95% 2 \
    -green final_K_crop_4.fits 0% 99.3% 2 \
    -blue final_J_crop_4.fits 0% 99.4% 2 \
    -out jpeg/r99.95_g99.3_b99.4_crop_4.jpg
[struct stat="OK"]
```

Acknowledgements

Montage is funded by the National Aeronautics and Space Administration's Earth Science Technology Office, Computational Technnologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology.). Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Pegasus is supported by NSF under grants ITR-0086044 (GriPhyN), ITR AST0122449 (NVO) and EAR-0122464 (SCEC/ITR). The Montage code is maintained by the NASA/IPAC Infrared Science Archive (IRSA).

References

Berriman, G. B. et al. 2002 in "Virtual Observatories," A. Szalay, ed. Proc of SPIE Volume 4846, 221

Berriman, G. B. et al. 2004 in "Optimizing Scientific Return for Astronomy through Information Technologies." P. J. Quinn and A. Bridger, eds. Proc of SPIE Vol 5393, 221

Bertin, E & Arnouts, S. 1996, A & AS, 117, 393.

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., & Kennedy, K. "Task Scheduling Strategies for Workflow-based Applications in Grids," Proceedings of CCGrid, Cardiff, UK, 2005.

Calabretta, M. R., & Greisen, E. W., A & A, 395, 1077, 2002.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G.,. Vahi, K, Berriman, G. B, Good, J., Laity, A., Jacob, J. C. & Katz, D. S., 2005, "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems," *Scientific Programming Journal*, vol. 13, pp. 219

Fruchter, A. S. & Hook, R. N. 2002 , PASP, 114, 144.

Katz, D. S., Berriman, G. B., Deelman, E., Good, J., Jacob, J. C., Kesselman, C., Laity, A. C., Prince, T. A., Singh, G. & M.-H. Su, 2005, "A Comparison of Two Methods for Building Astronomical Image Mosaics on a Grid", Proceedings of the 7th Workshop on High Performance Scientific and Engineering Computing (HPSEC-05)

Laity, A. C., Anagnostou, N., Berriman, G. B., Good, J. C., Jacob, J. C.,. Katz, D. S. & Prince, T. 2005, in Astronomical Data Analysis and Software Systems XIV, eds. P. L. Shopbell, M. C. Britton and R. Ebert, 34.

O'Rourke, J 1998, in Computational Geometry in C (Cambridge University Press), p220. (Chapter 7)

Singh, G., Deelman, E., Mehta, G., Vahi, K., Su, M.-H, Berriman, G. B., Good, J., Jacob, J. C., Katz, D. S., Lazzarini, A., Blackburn, K., and Koranda, S., 2005, "The Pegasus Portal: Web Based Grid Computing," Proceedings of 20th Annual ACM Symposium on Applied Computing, SAC, 2005.

Truong, H.-L. & Fahringer, T. 2005, "Online Performance Monitoring and Analysis of Grid Scientific Workflows," Proceedings of European Grid Conference 2005 (EGC), Amsterdam, The Netherlands.

Zhang, Y., Mandal, A, Casanova, H., Chien, A. A., Kee, Y.-S., Kennedy, K. and. Koelbel, C., 2006, "Scalable Grid Application Scheduling via Decoupled Resource Selection and Scheduling," Proceedings of CCGrid.

Useful Links

Montage. Jan 2007; http://montage.ipac.caltech.edu/ [Accessed 29 August, 2007]

Two Micron All-Sky Survey (2MASS) Dec 2006.; http://www.ipac.caltech.edu/2mass/ [Accessed 29 August, 2007]

Pegasus. Jun 2007; http://pegasus.isi.edu [Accessed 29 August, 2007]